COMP 2718: Command Line Parsing

By: Dr. Andrew Vardy

Adapted from the notes of Dr. Rod Byrne

Outline

- Command Line Parsing
- Desired Whitespace
- Multiline Commands
- Whitespace in Filenames

Command Line Parsing

The material presented here is not provided in the textbook.

The shell parses strings of text passed in to yield a set of **tokens**. Any **whitespace** counts as a **delimiter** between tokens:

Whitespace Space or tab characters (adjacent whitespace characters are considered one)

Delimiter Separating symbols in a string

Token The separate items between delimiters

On the command line, the first token is usually the command itself, while the rest are options or arguments. e.g.

```
$ echo hi there      # Two arguments

$ echo    hi  there  # Still two arguments
```

Desired Whitespace

So whitespace is effectively ignored when your command is parsed by `bash`. If you really want whitespace to be included, you must quote your arguments. A quoted argument is treated as one:

```
$ echo "hi     there" # One argument

hi     there
```

Remember that tabs are whitespace too:

```
$ echo -e "ARG1" "\tARG2\t" ARG3    ARG4

ARG1    ARG2     ARG3 ARG4
```

Multiline Commands

A newline character (‘\n’) marks the end of a line. If we want to use a command string spread over several lines we use a ‘\’ character at the end of each line when the command is to be continued on the next line. e.g.

```
$ echo hi \
> there \
> 2718!
hi there 2718!
```

If you don’t have a space before the ‘\n’ then the two intervening tokens will be combined as one:

```
$ echo howdy\
> folks
hodyfolks
```

Whitespace in Filenames

Spaces in filenames are allowed, but they can cause trouble. The following is legal:

```
$ echo "a message" > "the file"
```

```
$ cat the file
```

```
cat: the: No such file or directory
cat: file: No such file or directory
```

'\' should be used to escape the normal whitespace behaviour:

```
$ cat the\ file
a message
```